The proposed strengthened stream cipher at the heart of the protected communication method and system proposed is explained below. The system and method is designed for computational security and ease of use. Encryption strength is preferred over speed. The parts of the system used to create the improvement can be seen from the working algorithm submitted on microfiche with the patent application. If requested the software program itself will be sent on a CD with instructions to install, use, and test. The base ten algorithm that was submitted is also described in binary terms below, which is the preferred method. Both algorithms use the same transformation and transposition techniques as well as the means employed to strengthen the stream cipher. The improved results over other systems will be described in detail with each claim and tests proposed that any unbiased person with ordinary skill in the art would find convincing.

A detailed step by step encryption example in base 10 is also described so that the same principles applied in the preferred binary version of the cipher can be understood. This description coupled with the algorithm and working version of the software should be sufficient for one skilled in the art of encryption to know exactly how the software operates.

A list of obstacles that cryptanalysts must overcome to defeat the strengthened stream cipher will be provided. It will be demonstrated that cryptanalysts will not be able to discover information about the plaintext or user key used for encryption by any means.

I will begin with a reply to the Office Action Summary page by page as needed.

Specification Reply to Office Action Summary page 2

The One Time Pad (OTP) is the only provably secure encryption method to secure communications between devices. The OTP is a specialized stream cipher with severe limitations with the most significant being the distribution of sufficient one time pads for numerous encryption sessions. This makes the only perfectly secure cipher perfectly impractical. Since a practical solution to this distribution problem does not exist a more practical method to secure communications is proposed which offers the next highest level of security, computational security. Achieving this level of security means that all the computer power that exists now in the world, and what is expected to exist in the near future, would be inadequate to defeat the encryption if all of it was applied only to that task.

A strengthened stream cipher is the foundation of the proposed protected communication method and system and while it offers less than perfect security (like every cipher other than the OTP) it is computationally secure. It has been designed to withstand all the attacks commonly employed to crack a cipher including, but not limited to, the four types of attacks described by Schneier in Applied Cryptography, 2nd Edition on pages 5-6. They are listed as: 1)ciphertext-only attack 2) known-plaintext attack 3) chosen-plaintext attack and 4) adaptive-chosen-plaintext attack. This is with the algorithm fully disclosed to cryptanalysts and any amount of computer power applied. Only the user key must be kept secret. From page 152 of the 2nd edition of Applied Cryptography Schneier says,

"The security of a cryptosystem should rest in the key, not in the details of the algorithm. Assume that any cryptanalyst has access to all the details of your algorithm. Assume he has access to as much ciphertext as he wants and can mount an intensive ciphertext-only attack. Assume that he can mount a plaintext attack with as much data as he needs. Even assume that he can mount a chosen-plaintext attack. If your cryptosystem can remain secure, even in the face of all that knowledge, then you've got something."

Unlike the OTP the proposed communication method and system is infinitely more practical since the pads generated to combine with plaintext are made by the sending device during encryption. The only thing a user must do is enter a key once into the device for millions of secure encryption sessions. The same master key used in the device to send must also be entered once into each device to decrypt messages received. The pads come from the user key chosen which is treated as a bit ring. Bits from the ring are selected to combine with the plaintext bit string to disguise it using a new one way function called EndPoint Permutation. The bits used for this control are communicated in shorthand form as a subkey that is thoroughly interspersed by numbered positions as bits into the c-text bit string in fixed block sizes.

The proposed method removes the need for ongoing, real-time, synchronization between devices. Sending and receiving devices act independently of one another and there is no limit to the number of devices that may be used. Synchronization is a part of the decryption process with each block of cipher text (c-text) received. Subkeys (controlling bits) cannot be discovered since they are random, and can even be perfectly random if

desired, and are completely mixed into a variety of numbered positions of each c-text block. The positions of the bits that make up the subkey are determined by the user key and vary in position, as they vary in value, with each block of c-text transmitted. The user key can be any length that communicating devices have enough memory to store leaving enough memory to process. A minimum key length and an easy way to generate it is recommended to users. A 4,000 character key made up of c-text is recommended for the less preferred base 10 version of software. A minimum key of c-text is recommended for the binary version that is a little longer, around 10k, and may be as long as the user desires for increased security.

EndPoint Permutation is superior to public key encryption in that the one way function is a process that cannot be cracked by a single mathematical operation. The prime factors that were used to create a public key can be discovered by finding a single prime number that will divide it without a remainder. Many popular encryption methods and systems rely upon no shortcuts existing for doing this quickly. Schneier discusses the dangers of fast factoring discovery on pg 435 of the 2nd edition of Applied Cryptography. Many popular encryption algorithms rely upon factoring remaining difficult. EndPoint permutation is immune to fast factoring. Once used the resulting permutation cannot be reversed or cracked by a single mathematical operation. EndPoint Permutation also has the advantage of being able to effectively rearrange a bit string of any size using a small number of bits. The proposed method and system is unique and this is easily proven. There is no record of this superior one way function being used in a protected communication method and system before. While researching published literature no

record of EndPoint Permutation on prime and or non-prime strings has been found although its usefulness is obvious to one skilled in the art. Rearranging a sequence with a partial EndPoint Permutation procedure before a full one will greatly increase the number of permutations possible and approach the factorial of that number of elements in a sequence, the max number. A random section of ring elements may also be removed before EndPoint Permutation is applied resulting in a higher number of unique permutations of the original large bit ring.

The proposed method and system is also superior to popular pseudo random number generators (prng's) since the random number settings used, even if known, do not reveal the bit string generated as it does with prng's. To know the settings of a prng is to know the output. The same is not true of the proposed method and system since they are generated from the user key which is kept secret. All that is required is for the user to enter a relatively random key of a recommended size. From that key enough unique pads are generated within the device for millions of encryption sessions. The high quality keystream generated by the sending device cannot be determined by knowing the subkey used as it is in other pseudo random number generators. That is because the heart of the one way function is determined by the user key.

EndPoint Permutation, a superior one way function

To clearly understand how the proposed one way function works in a cipher see the examples below that vary in size. The preferred ring size, or bit pool, would usually have about 64,000 bits. The user may have more bit pools as desired for added security. See

the following identical 7 bit strings with EndPoint Permutation applied using different random numbers. Start at the element position specified (left to right) and then jump the number given (left to right) treating the bit string as a ring. See the following examples of two short prime rings to see how EndPoint Permutation rearranges two identical bit rings using different start and jump numbers.

3 1 6 4 2 7 5 1 2 3 4 5 6 7
0 1 0 1 1 0 0 with a starting point of 2 and jump of 3 becomes 1 1 0 1 0 0 0

2 5 1 4 7 3 6 1 2 3 4 5 6 7
0 1 0 1 1 0 0 with a starting point of 3 and jump of 5 becomes 0 0 0 1 1 0 1

The same procedure is applied below to the four bit strings after ghost bits (G) are added to make the number of elements prime. The ghost bits are then removed. They could also be used to trigger a logical operation such as XOR or XNOR or other logical operations, or combinations of operations, as determined by the key or subkey if desired as a strengthening factor instead of simply removing them.

3 1 6 4 2 7 5 1 2 3 4 5 6 7
0 0 11 becomes 0 0 1 1 G G G with a start of 2 and jump of 3 yields 0G01G1G = 0011

2 5 1 4 7 3 6 1 2 3 4 5 6 7
0 0 1 1 becomes 0 0 1 1 G G G with a start of 3 and jump of 5 yields 10G10GG = 1010

For a longer bit string permutation see the following example:

Original ring of 37 key bits:

1011110010001101011001111100010101000

From the original ring started at 36 with jump of 2 gives...

01110101001011000000011000111011011 1

The number of unique permutations using the proposed method is equal to the number of elements multiplied by one less than that number. So a set of 37 elements can be rearranged in 37 x (37-1) or 1,332 ways using one simple round of EndPoint Permutation. As long as the number of elements are prime and the jump size does not equal the number of elements undergoing EndPoint Permutation a bit will be selected only once as the ring is circled until all the elements have been selected resulting in a rearranged bit sequence.

It is surprising to see that any number of bits can easily be rearranged using this one way function, including bit strings that are not prime, using large or small numbers to start and jump around the ring sequence. The number of permutations can also be increased by a partial EndPoint Permutation of a bit string before doing a full EndPoint Permutation. Bits harvested would simply be put somewhere in the string after the selected bits were deleted. Doing this partial procedure one or more times would be a quick way to greatly increase the number of permutations possible since the original string would be changed significantly before the full procedure was applied. Removing a small subset of elements before the procedure also increases the possible number of permutations.

Analysis of many large bit strings after EndPoint Permutation has been applied to them do not easily show the original bit string they originated from, especially after applying a final EndPoint Permutation procedure to them with different start and jump numbers. Or using matrices determined by the user key for transposition after this stage would greatly

complicate the cryptanalysts task. Determining the original bit string sequence by examining a large number of strings after EndPoint Permuation with random start and jump numbers is very difficult. When other strengthening factors are added to this difficulty those skilled in the art of encryption will have to admit that cracking the cipher is a "hard" problem as Schneier describes it in pages 29 and 44-45 of Applied Crypto 2nd Edition "...'hard' is defined as something like: It would take millions of years to compute x from f(x), even if all the computers in the world were assigned to the problem."

Unbiased cryptanalysts will find that the proposed communication method and system is a hard problem for codebreakers to solve. The quality of the cipher text it produces should be examined. Standard statistical tests may be applied to the base 10 algorithm by taking a 5,000 character c-text message and converting it to digits using the tools pull down menu to choose "cipher to digits" in the program. The 10,000 digit message can then be converted to 20,000 bits by replacing the even digits with zero's and the odd digits with 1's (Using "Find and Replace" in Word does this nicely) resulting in a binary sequence for empirical analysis. Any plaintext message may be encrypted for testing provided a 4,000 character user key composed of c-text is used, as recommended, in the base 10 algorithm. The degree of randomness of the binary stream produced rivals that of the highest quality prng's now popular. In comparing 5,000 c-text characters against 5,000 digits of pi both appear equally random. It is surprising to see that the statistical tests for c-text produced (regardless of text coded) show better randomness than pi once both are converted to bits in the same way. In fact, the proposed communication method and system in the preferred binary form can stand alone as a superior prng (pseudo